PROYECTO TRANSPORTE ANGULAR

Comenzaremos mostrando la estructura de los esquemas de la API Rest. Tenemos dos colecciones, una de trabajadores y una de vehiculos.

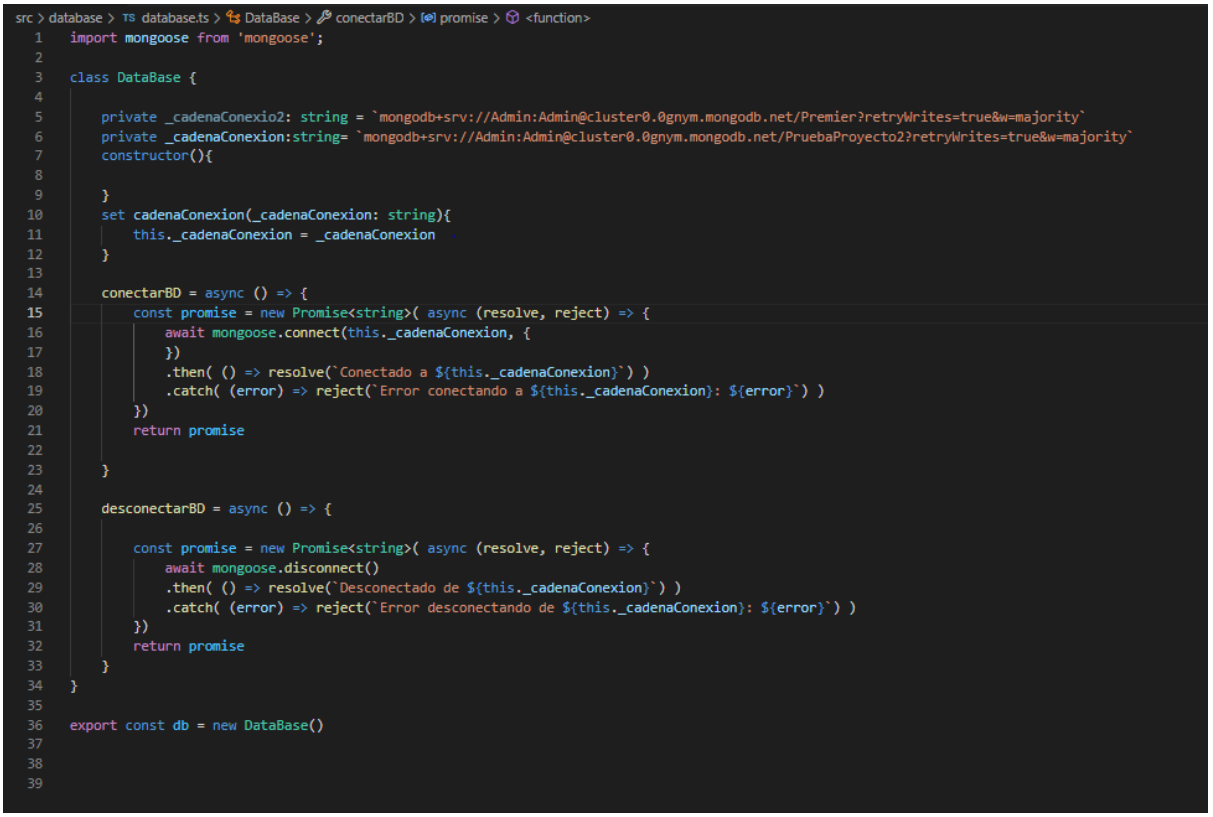
model > TS trabajadores.ts > ...

```ts
import { Schema, model } from "mongoose";

const TrabajadorSchema = new Schema({
    DNI: String,
    nombre: String,
    apellidos: String,
    fechaNac: Date,
    salHora: Number,
    cargo: String,
    tipoT: String,
    especialidad: Array,
    ubicacion: String,
    licencias: Array,
    incidencias: Array
})

export type iConductor = {
    DNI: String | null,
    nombre: String | null,
    apellidos: String | null,
    fechaNac: Date | null,
    salHora: Number | null,
    cargo: String | null,
    tipoT: String | null,
    licencias: Array<string> | null,
    incidencias: Array<string> | null
}

export type iMecanico = {
    DNI: String | null,
    nombre: String | null,
    apellidos: String | null,
    fechaNac: Date | null,
    salHora: Number | null,
    cargo: String | null,
    tipoT: String | null,
    especialidad: Array<string> | null,
    ubicacion: String | null
}

export type iTrabajador = {
    DNI: String,
    nombre: String,
    apellidos: String,
    fechaNac: Date,
    salHora: Number,
    cargo: String,
    licencias: [],
    incidencias: []
}

export const Trabajadores = model('trabajador', TrabajadorSchema  )
```

model > TS vehiculos.ts > iAutobus

```ts
import { Schema, model } from "mongoose";

const VehiculoSchema = new Schema({
    matricula: String,
    numPlazas: Number,
    fechaInicio: Date,
    pagoTarjeta: Boolean,
    trabajadores: Array,
    tipoT: String,
    tipoTren: String,
    estaciones: Array,
    bano: Boolean,
    numPlantas: Number
},
)

export type iAutobus = {
    matricula: String | null,
    numPlazas: Number | null,
    fechaInicio: Date | null,
    pagoTarjeta: Boolean | null,
    trabajadores: Array<string> | null,
    bano: Boolean | null,
    numPlantas: Number | null,
    tipoT: String | null,
}

export type iTren = {
    matricula: String | null,
    numPlazas: Number | null,
    fechaInicio: Date | null,
    pagoTarjeta: Boolean | null,
    trabajadores: Array<string> | null,
    tipoTren: String | null,
    estaciones: Array<string> | null,
    tipoT: String | null,
}

export const Vehiculos = model('vehiculos', VehiculoSchema  )
```

Utilizaremos una base de datos de mongo atlas



La API rest contiene GET, PUT, POST y DELETE de los que tenemos los siguientes ejemplos

```
// Recibe un documento equipo en el body con los campos indicados aquí
private postConductor = async (req: Request, res: Response) => {
    const { DNI, nombre, apellidos, fechaNac, salHora, cargo, tipoT, licencias, incidencias } = req.body
    await db.conectarBD()
    dSchemaConductor = {
        DNI: DNI,
        nombre: nombre,
        apellidos: apellidos,
        fechaNac: fechaNac,
        salHora: salHora,
        cargo: cargo,
        licencias: licencias,
        incidencias: incidencias,
        tipoT: tipoT
    }
    const oSchema = new Trabajadores(dSchemaConductor)
    await oSchema.save()
        .then( (doc: any) => res.send(doc))
        .catch( (err: any) => res.send('Error: '+ err))
    await db.desconectarBD()
}
```

```
private getTrabajadores = async (req:Request, res: Response) => {
    await db.conectarBD()
    .then( async ()=> {
        const j = await Trabajadores.find({})
        res.json(j)
    })
    .catch((mensaje) => {
        res.send(mensaje)
    })
    await db.desconectarBD()
}
```

```
private updateVehiculo = async (req: Request, res: Response) => {
    const { matricula } = req.params
    const {  tipoTransp, numPlazas, fechaInicio, conductores, trayecto,
             combustible, pagoT } = req.body
    await db.conectarBD()
    await Vehiculos.findOneAndUpdate({
        matricula: matricula
    },{
        tipoTransp: tipoTransp,
        numPlazas: numPlazas,
        fechaInicio: fechaInicio,
        conductores: conductores,
        trayecto: trayecto,
        combustible: combustible,
        pagoT: pagoT
    },{
        new: true, // para retornar el documento después de que se haya aplicado la modificación
        runValidators:true
    }
    )
        .then( (doc: any) => res.send(doc))
        .catch( (err: any) => res.send('Error: '+ err))
    await db.desconectarBD()
}
```

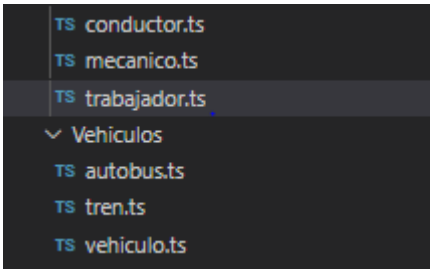
```
    private deleteTrabajador = async (req: Request, res: Response) => {
        const { DNI } = req.params
        await db.conectarBD()
        await Trabajadores.findOneAndDelete(
                { DNI: DNI}
            )
            .then( (doc: any) => {
                    if (doc == null) {
                        res.send(`No encontrado`)
                    }else {
                        res.send('Borrado correcto: '+ doc)
                    }
            })
            .catch( (err: any) => res.send('Error: '+ err))
        db.desconectarBD()
    }

    private deleteVehiculo = async (req: Request, res: Response) => {
        const { matricula } = req.params
        await db.conectarBD()
        await Vehiculos.findOneAndDelete(
                { matricula: matricula}
            )
            .then( (doc: any) => {
                    if (doc == null) {
                        res.send(`No encontrado`)
                    }else {
                        res.send('Borrado correcto: '+ doc)
                    }
            })
            .catch( (err: any) => res.send('Error: '+ err))
        db.desconectarBD()
    }
```

Las rutas al completo serían las siguientes.

```
71
72      misRutas(){
73         // POST
74          this._router.post('/conductor', this.postConductor),
75          this._router.post('/mecanico', this.postMecanico),
76          this._router.post('/tren', this.postTren),
77          this._router.post('/autobus', this.postAutobus),
78          // GET
79          this._router.get('/vehiculos', this.getVehiculos),
80          this._router.get('/trenes', this.getTrenes),
81          this._router.get('/autobuses', this.getAutobuses),
82          this._router.get('/trabajadores', this.getTrabajadores),
83          this._router.get('/conductores', this.getConductores),
84          this._router.get('/mecanicos', this.getMecanicos),
85          this._router.get('/trabajador/:DNI', this.getTrabajador),
86          this._router.get('/vehiculo/:matricula', this.getVehiculo),
87          // PUT
88          this._router.put('/updateVehiculo/:matricula', this.updateVehiculo),
89          this._router.put('/updateTren/:matricula', this.updateTren),
90          this._router.put('/updateAutobus/:matricula', this.updateAutobus),
91          this._router.put('/updateTrabajador/:DNI', this.updateTrabajador),
92          this._router.put('/updateConductor/:DNI', this.updateConductor),
93          this._router.put('/updateMecanico/:DNI', this.updateMecanico),
94          // DELETE
95          this._router.delete('/deleteTrabajador/:DNI', this.deleteTrabajador),
96          this._router.delete('/deleteVehiculo/:matricula', this.deleteVehiculo)
97      }
98  }
99
```

A continuación se mostrará la documentación de la aplicación de angular.

Tenemos las siguientes clases.



Siendo vehíco y trabajador las superclases.

En el archivo app.routing.module contenemos todas las rutas del proyecto

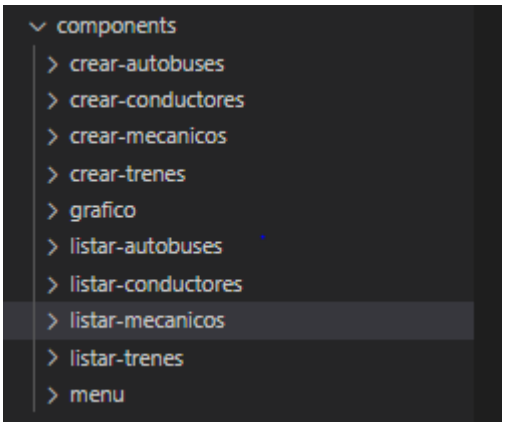
```
const routes: Routes = [
  { path: 'listar-tren', component: ListarTrenesComponent},
  { path: 'crear-tren', component: CrearTrenesComponent},
  { path: 'editar-tren/:matricula', component: CrearTrenesComponent},
  { path: 'listar-autobus', component: ListarAutobusesComponent},
  { path: 'crear-autobus', component: CrearAutobusesComponent},
  { path: 'editar-autobus/:matricula', component: CrearAutobusesComponent},
  { path: 'listar-mecanico', component: ListarMecanicosComponent},
  { path: 'crear-mecanico', component: CrearMecanicosComponent},
  { path: 'editar-mecanico/:DNI', component: CrearMecanicosComponent},
  { path: 'listar-conductor', component: ListarConductoresComponent},
  { path: 'crear-conductor', component: CrearConductoresComponent},
  { path: 'editar-conductor/:DNI', component: CrearConductoresComponent},
  { path: 'grafico', component: GraficoComponent},
  { path: '', component: MenuComponent},
  { path: '**', redirectTo: '', pathMatch: 'full'}
];
```

En el archivo service realizamos la conexión entre la API y la aplicación angular.

Definiendo la url de la API y añadiendo la ruta necesaria a la misma.

```
@Injectable({
  providedIn: 'root'
})
export class TransporteService {
  private url = 'http://localhost:3000';
  constructor(private http: HttpClient) { }


  // GET
  getTrenes(): Observable<any> {
    const url = `${this.url}/trenes`;
    return this.http.get(url);
  }

  getAutobus(): Observable<any> {
    const url = `${this.url}/autobuses`;
    return this.http.get(url);
  }

  getConductores(): Observable<any> {
    const url = `${this.url}/conductores`;
    return this.http.get(url);
  }

  getMecanicos(): Observable<any> {
    const url = `${this.url}/mecanicos`;
    return this.http.get(url);
  }

  getTrabajador(DNI: string): Observable<any> {
```

La aplicación contiene una serie de componentes dentro de la carpeta components.



Los componentes son de varios tipos y vamos a ver los siguientes ejemplos.

```
    trabajadores: ['', Validators.required],
    bano: ['', Validators.required],
    numPlantas: ['', Validators.required],
    tipoT: ['', Validators.required],
  })
  this.matricula = this.aRouter.snapshot.paramMap.get('matricula');
}

ngOnInit(): void {
  this.editarAutobus();
}


postAutobus() {
  const Autobus: iAutobus = {
    matricula: this.autobusForm.get('matricula')?.value,
    numPlazas: this.autobusForm.get('numPlazas')?.value,
    fechaInicio: this.autobusForm.get('fechaInicio')?.value,
    pagoTarjeta: this.autobusForm.get('pagoTarjeta')?.value,
    trabajadores: this.autobusForm.get('trabajadores')?.value,
    bano: this.autobusForm.get('bano')?.value,
    numPlantas: this.autobusForm.get('numPlantas')?.value,
    tipoT: this.autobusForm.get('tipoT')?.value,
  }

  if(this.matricula !== null){
    this.transporteService.updateAutobus(this.matricula, Autobus).subscribe()
      this.toastr.info('El autobus fue actualizado correctamente');
      this.autobusForm.reset()
  } else {
    this.transporteService.postAutobus(Autobus).subscribe()
    this.toastr.success('Añadido correctamente');
    this.autobusForm.reset()
  }
```

```
import { Autobus } from '../../models/Vehiculos/autobus';
import { TransporteService } from '../../services/transporte.service';

@Component({
  selector: 'app-listar-autobuses',
  templateUrl: './listar-autobuses.component.html',
  styleUrls: ['./listar-autobuses.component.css']
})
export class ListarAutobusesComponent implements OnInit {

  listAutobuses: Autobus[] = [];
  constructor(private TransporteService: TransporteService,
              private toastr: ToastrService
    ) {}

  ObtenerAutobus() {
    this.TransporteService.getAutobus().subscribe((data) => {
      this.listAutobuses= data;
    });
  }

    deleteAutobus(matricula: any) {
      this.TransporteService.deleteVehiculo(matricula).subscribe(data => {
        this.toastr.error('El autobus fue eliminado correctamente')
        this.ObtenerAutobus();
      })
    }



  ngOnInit(): void {
    this.ObtenerAutobus();
  }
}
```

```
op > components > menu > menu.component.html > nav.navbar.navbar-expand-lg
<nav class="navbar navbar-expand-lg" style="background-color: black;">
  <a class="navbar-brand" href="#" routerLink="/">TRANSPORTE</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="#" routerLink="/listar-tren">TREN<span class="sr
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#" routerLink="/listar-autobus">AUTOBUS</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#" routerLink="/listar-conductor">CONDUCTOR</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#" routerLink="/listar-mecanico">MECANICOS</a>
      </li>
    </ul>
  </div>
</nav>
```

```
    this.getSalario()
  }

  //SeriesOptionsType
  getSalario() {
    this._transporteService.getMecanicos()
    .subscribe(
      (result: any) => {
        this.listMecanicos = result.map((mecanico: any) => {
          return new Mecanico(mecanico.DNI, mecanico.nombre, mecanico.apellidos, mecanico.fechaNac, mecanico.sa
        });
        console.log(this.listMecanicos)

// Creamos los objetos y usamos un método para representar el valor devuelto
        const dataSeries = this.listMecanicos.map((x: Mecanico) => x.DNI);
        const dataCategorias = this.listMecanicos.map((x: Mecanico) => x.salHora);
        if(dataSeries!=undefined && dataCategorias !=undefined && this.chartOptions.series!=undefined && this.c
        this.chartOptions.series[0]["data"] = dataCategorias;
        this.chartOptions.xAxis["categories"] = dataSeries;
        console.log(this.chartOptions.series[0]["data"])
        console.log(this.chartOptions.xAxis["categories"])
        this.chartOptions.title["text"] = "DNI";
        this.chartOptions.series["name"] = "SALARIO HORA"
        Highcharts.chart("miGrafico01", this.chartOptions);
        }
      },
      error => console.log(error)
    );
  }
}
```